Client Rental Database

Property Management System built using MySQL

By: Egshiglen Enkhbayar

Tools: MySQL Workbench · ERD Modelling · SQL Queries

1. Project Overview

The *Client Rental Database* project models a property rental management system designed for a real estate company. The goal was to design, normalize, and implement a relational database that efficiently stores client, property, owner, and rental information. The database enables quick access to client-property relationships and provides business insights through analytical SQL queries.

---

2. Objectives

- Design a relational database that minimizes redundancy and improves consistency.
- Apply normalization techniques (1NF → 3NF) to structure data efficiently.
- Model relationships between entities using ERD diagrams (Chen and Crow's Foot notations).
- Populate the database with dummy data representing real-world scenarios.
- Implement SQL queries to analyze client behavior, property ownership, and rent trends.

---

3. Database Design Process

3.1 Normalization

The initial unnormalized dataset was transformed step-by-step into normalized tables:

First Normal Form (1NF):
Removed repeating and multivalued attributes by creating separate tables for Clients and RentInfo.

Second Normal Form (2NF):
Eliminated partial dependencies by introducing a new table, Properties, separating property information from rental logs.

Third Normal Form (3NF):
Removed transitive dependencies by creating an Owners table, ensuring that owner details are only stored once and linked via foreign keys.

3.2 Final Entities

- **Clients (Client_No, Client_Name)**
- **Owners (Owner_No, Owner_Name)**
- **Properties (Property_No, Property_Address, Monthly_Rent, Owner_No)**
- **RentInfo (Client_No, Property_No, Rent_Start, Rent_Finish)**

Each entity uses primary and foreign key constraints to maintain referential integrity.

---

4. Implementation

The database was implemented in **MySQL Workbench** using DDL and DML statements.

Example Commands

- **CREATE DATABASE ClientRentalDB;**
- **CREATE TABLE Clients(...);**
- **INSERT INTO Clients VALUES (...);**

All tables were tested using DESCRIBE and SELECT statements to verify data structure and relationships.

Key Relationships

- One client can rent multiple properties.
- One owner can own multiple properties.
- One property belongs to one owner.

---

5. Analytical SQL Queries

Ten SQL queries were implemented to extract insights and verify relational integrity.

#	Description	SQL Concept Used
1	Retrieve all clients with their rented properties	JOIN
2	List properties rented by clients whose names begin with 'D'	LIKE, JOIN
3	List clients renting within a specific date range	WHERE, BETWEEN
4	Calculate total monthly rent per client	SUM, GROUP BY
5	Find the owner of a specific property	JOIN, WHERE
6	Count total properties per owner	COUNT, GROUP BY
7	Identify owners with multiple properties	HAVING, GROUP BY
8	Compute total annual rent per client (ascending order)	ORDER BY, SUM
9	Find the client paying the highest rent	MAX, subquery
10	List properties with rent above average	AVG, subquery

---

6. Results & Insights

- Determined which clients rent the most properties and pay the highest rent.
- Identified top property owners and their property counts.
- Calculated average and total rent values for data analysis.
- Demonstrated correct one-to-many and many-to-one relationships.

These results reflect the database's ability to support decision-making in real estate management scenarios.

---

7. Tools & Technologies

Tool	Purpose
MySQL Workbench	Database creation and ERD visualization
SQL	Data definition, insertion, and analytical queries
Random Name Generator	Generated dummy client and owner names
Excel / MySQL Query Output	Used for result interpretation and screenshots

---

8. Critical Analysis

This project highlighted the importance of normalization and relational integrity.

- **Strengths:** Eliminated redundancy, maintained consistency, and improved scalability.
- **Challenges:** Designing logical keys and ensuring correct joins across multiple entities.
- **Learning Outcome:** Strong understanding of how SQL joins, aggregation, and subqueries provide meaningful insights for business systems.

In a real-world property rental company, such a database could streamline rent tracking, client management, and owner reporting.

---

9. Conclusion

The *Client Rental Database* demonstrates the complete lifecycle of database design — from normalization and ERD modeling to implementation and analysis.
Through this project, I gained hands-on experience in relational schema design, data integrity management, and SQL query optimization.

This database can easily be expanded to include new features such as payment tracking, maintenance logs, or client feedback systems.

Table of Contents

1.1 Databases Part 1

1. Convert the unnormalized data into first, second, and third normalized tables

Note: In this section, only 2 clients have been used as an example of the normalization and in the next sections, more data will be added to the database.

1. **First Normal Form (1NF):**

The step-by-step procedure for converting the unnormalized table into 1NF. Creating the normalized resultant 1NF tables.

UNF(Unnormalized form)

Client_No	Client_Name	Property_No	Property_Address	Rent_start	Rent_finish	Monthly_rent	Owner_No	Owner_Name
CR-67	John Kay	PG4	6 Lawrence, Glasgow.	1-Jul-17	31-Aug-18	350	C040	Tina
CR-67	John Kay	PG16	5 Nova St, Glasgow.	1-Sep-18	1-Sep-19	450	C093	Tom
CR-68	Aline Stewart	PG4	6 Lawrence, Glasgow.	1-Sep-16	10-Jun-17	350	C040	Tina
CR-68	Aline Stewart	PG36	2 Monar St,Glasgow.	10-Oct-17	1-Dec-18	375	C093	Tom
CR-68	Aline Stewart	PG16	5 Nova St, Glasgow.	1-Nov-19	10-Aug-20	450	C093	Tom

UNF to 1NF (1st normal form)

1NF has no multivalued attributes

a. Identifying the multivalued attributes

Clients

- Client_No(PK)
- Client_name

(Multivalued attributes)

- Property_No
- Property_Address
- Rent_start
- Rent_finish
- Monthly_rent
- Owner_No
- Owner_Name

b. Removing the repeating attributes and creating a new entity, giving it a meaningful name called RentInfo.

RentInfo

- Property_No
- Property_Address
- Rent_start
- Rent_finish
- Monthly_rent
- Owner_No
- Owner_Name

c. Identifing a primary key for this new entity

RentInfo

- Property_No(PK)
- Property_Address
- Rent_start
- Rent_finish
- Monthly_rent
- Owner_No
- Owner_Name

d. Taking the primary key from the first table "Clients" and use it as a foreign key so that both tables (Clients and RentInfo) are linked to each other

RentInfo

- Client_No(FK)
- Property_No(PK)
- Property_Address
- Rent_start
- Rent_finish
- Monthly_rent
- Owner_No
- Owner_Name

e. A composite key (two or more attributes together form a composite key that can uniquely identify a tuble in a table)

Composite key = Clients + Property_No

f. We have two different entities called Clients and Rent

Clients

Client_No(PK)	Client_Name
CR-67	John Kay
CR-68	Aline Stewart

RentInfo

Client_No(FK)	Property_No(PK)	Property_Address	Rent_start	Rent_finish	Monthly_rent	Owner_No	Owner_Name
CR-67	PG4	6 Lawrence, Glasgow.	1-Jul-17	31-Aug-18	350	C040	Tina
CR-67	PG16	5 Nova St, Glasgow.	1-Sep-18	1-Sep-19	450	C093	Tom
CR-68	PG4	6 Lawrence, Glasgow.	1-Sep-16	10-Jun-17	350	C040	Tina
CR-68	PG36	2 Monar St,Glasgow.	10-Oct-17	1-Dec-18	375	C093	Tom
CR-68	PG16	5 Nova St, Glasgow.	1-Nov-19	10-Aug-20	450	C093	Tom

The important to ensure that the table adheres to 1NF

First normalization helps to distinguish multivalued attributes from the unnormalized table and eliminates the repeating attributes. It also ensures that all values in each column are atomic / indivisible. This step helps remove any multivalued or composite data, and the result can be normalized in the later stages (2NF, 3NF). Without doing it, the repeating and multivalued attributes could fail the normalization and hard to create, insert, update data into the table because it's complicated. Every data in UNF table is mixed so it would be hard to categorize and manipulate the data.

Advantages:

- Creates the foundation data with consistency and no ambiguity
- No duplication and redundancy making the data easier for the query

2. **Second Normal Form (2NF):**

1NF to 2NF (2nd normal form)

2NF has no partial key dependencies

a. First, identifying the composite key

Composite key = Clients + Property_No

b. Then, looking for attributes which rely on only one of the composite keys to exist.

In this scenario, in the entity RentInfo the attribute "Property_Address", Monthly_Rent_ is dependent solely on the prime attribute "Property_No", indicating a partial dependence. On the other hand, the non-prime attributes are "Rent_start", "Rent_finish".

RentInfo

- Client_No(FK)
- Property_No(PK)
- Property_Address
- Rent_start
- Rent_finish
- Monthly_rent
- Owner_No
- Owner_Name

c. Removing the partial dependent attributes and creating a new entity, giving it a name called Properties

Properties

- Property_No
- Property_Address
- Monthly_Rent
- Owner_No
- Owner_name

d. Identify a primary key for this new entity

Properties

- Property_No(PK)
- Property_Address
- Monthly_Rent
- Owner_No
- Owner_name

e. Since the new key is a primary key in a new entity(Properties), make it a foreign key in the original entity (RentInfo)

RentInfo

- Client_No(FK)
- Property_No(FK)
- Rent_start
- Rent_finish

Properties

- Property_No(PK)
- Property_Address
- Monthly_Rent
- Owner_No
- Owner_name

f. We have three different entities Clients, RentInfo, Properties

RentInfo

Client_No(FK)	Property_No(FK)	Rent_start	Rent_finish
CR-67	PG4	1-Jul-17	31-Aug-18
CR-67	PG16	1-Sep-18	1-Sep-19
CR-68	PG4	1-Sep-16	10-Jun-17
CR-68	PG36	10-Oct-17	1-Dec-18
CR-68	PG16	1-Nov-19	10-Aug-20

Properties

Property_No(PK)	Property_Address	Monthly_rent	Owner_No	Owner_Name
PG4	6 Lawrence, Glasgow.	350	C040	Tina
PG16	5 Nova St, Glasgow.	450	C093	Tom
PG36	2 Monar St,Glasgow.	375	C093	Tom

Clients

Client_No(PK)	Client_Name
CR-67	John Kay
CR-68	Aline Stewart

The benefits of achieving 2NF, specifically in the context of the client rental form

By achieving 2NF, it eliminates any partial dependencies which makes sure that all non-prime attributes are fully dependent on the primary key.

Advantages:

- Prevents redundancy in partial attributes
- The data can be distinguish between Rent information and property information. If you want to check when is CR-67 client is checking in and checking out you just use the RentInfo table but which property they're renting you go to properties table. It's helpful to find information based on what specific information you're looking for.
- By separating data that belongs to different entities, it simplifies the data management
- Saves a lot of time for query management and analysis

3. **Third Normal Form (3NF):**

The steps involved in transforming the table from 2NF to 3NF. Creating the normalized resultant 3NF tables.

2NF to 3NF (3rd normal form)

3NF = No non-key dependencies or transitive dependencies

a. Examining at all the entities produced so far such as "Clients", "RentInfo", and "Properties" and identifyong any non-prime attributes which rely on any other non-prime attributes or transitive dependencies.

In this scenario, within the entity "Properties", the non-primitive attribute "Owner_Name" is dependent on the non-primitive attribute "Owner_No" , demonstrating transitive dependencies.

Properties

- Property_No(PK)
- Property_Address
- Monthly_Rent
- Owner_No
- Owner_name

b. Removing the transitive dependencies attributes "Owner_Name" and creating a new entity, giving it a name called "Owners".

Owners

- Owner_No
- Owner_name

c. Identify a primary key for this new entity

Owners

- Owner_No(PK)
- Owner_name

d. Since the new key is a primary key in a new entity (Owners), make it a foreign key in the original entity(Properties)

Properties

- Property_No(PK)
- Property_Address
- Monthly_Rent
- Owner_No(FK)

e. We have four different entities

Clients

Client_No(PK)	Client_Name
CR-67	John Kay
CR-68	Aline Stewart

RentInfo

Client_No(FK)	Property_No(FK)	Rent_start	Rent_finish
CR-67	PG4	1-Jul-17	31-Aug-18
CR-67	PG16	1-Sep-18	1-Sep-19
CR-68	PG4	1-Sep-16	10-Jun-17
CR-68	PG36	10-Oct-17	1-Dec-18
CR-68	PG16	1-Nov-19	10-Aug-20

Properties

Property_No(PK)	Property_Address	Monthly_rent	Owner_No(FK)
PG4	6 Lawrence, Glasgow.	350	C040
PG16	5 Nova St, Glasgow.	450	C093
PG36	2 Monar St,Glasgow.	375	C093

Owners

Owner_No(PK)	Owner_Name
C040	Tina
C093	Tom

The significance of attaining 3NF in the database schema:

- 3NF helps to separate non-prime attributes from the original table, removes transitive dependencies
- Gives assurance for non-prime attributes depending only on the primary key and not with the other non-prime attributes.
- By separating them it creates a new entities that contains information which could be used for only specific reasons like if someone's renting a room and if you want to know only the address, you don't need owner's name. So you use the data from Properties table, not owner's table. Only if you want to know the owner you use the owner's table.
- Avoids redundancy in indirect dependencies
- Helps improve the data integrity and removes anomalies
- Making the data design more efficient and simple for data management.

The advantages and disadvantages of normalization in the context of this database design client rental form.

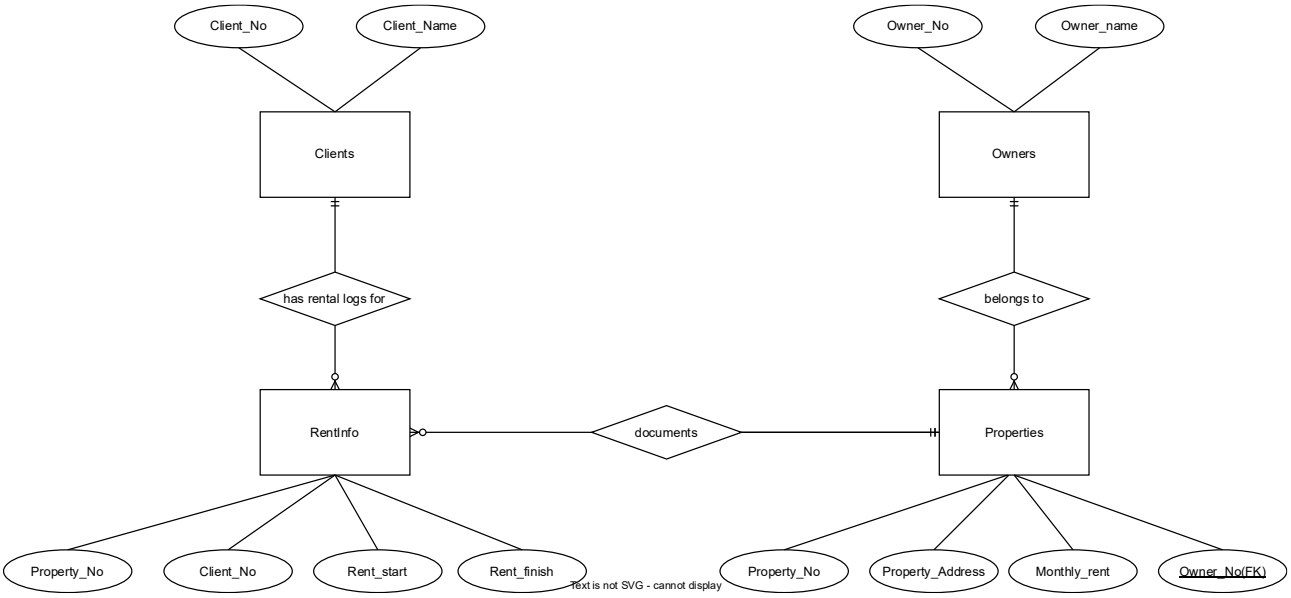
The advantages were that using normalization in client rental form now they have organized, separated data to work on such as Clients, RentInfo, Properties, Owners. By separating and creating these entities, the renting company can search and organize the information much faster than unnormalized form. It saves time and creates efficiency by it's clear, understandable design. Easy to add or remove information. If the renting company wants to know how many owners they have they just use the owners table. Or how many properties John rented in the last year they can look up on RentInfo table. And there's no repeating attributes which helps it to making it simpler.

The disadvantages could be that to find specific information like Client's Name and you don't know which entity it is. It could be a problem. Or to add another entity but we have to change the entire design, also can be a problem.

How normalization contributes to the overall efficiency and effectiveness of the database design.

If we look at the ClientRental data unnormalized form, everything seems confusing and hard to understand, because it's all in one entity. But if we do normalization it creates entities for every specific information. If we want to change or look for information, we don't need to waste time by going over everything, we just need to find the right table. The normalizations design makes the data management simpler and making it simpler means, making it easier to use and understand for whomever is using that database for queries and data manipulation.

2.Provide the ERD using CHEN notation



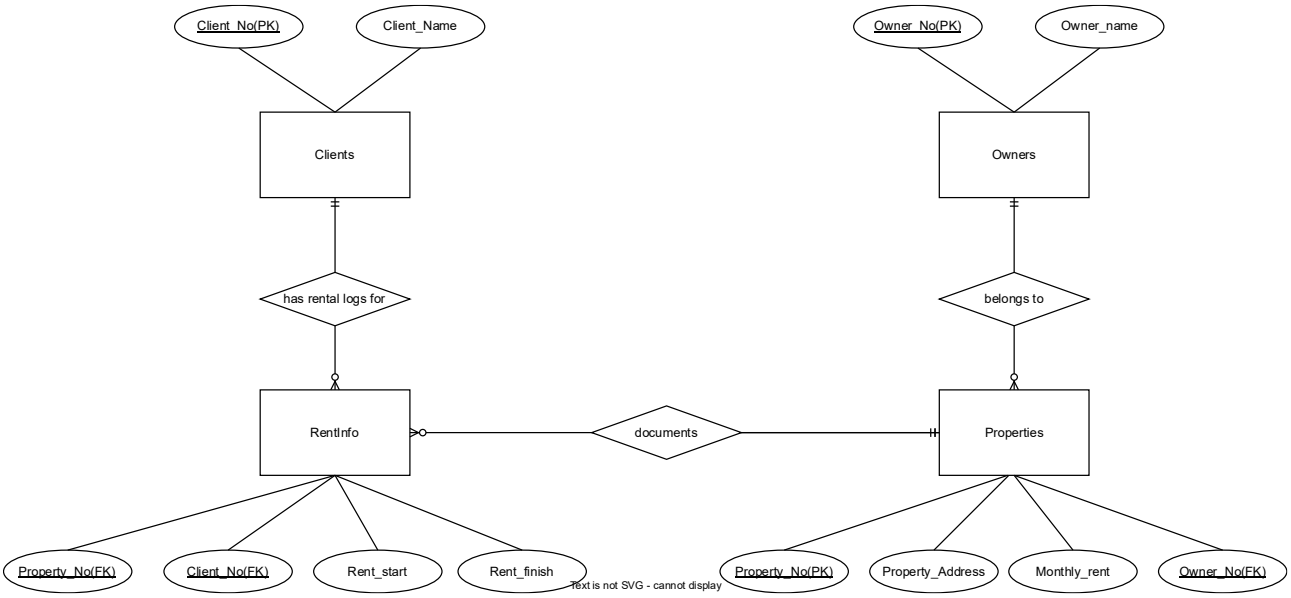
Chen notation relationships:

Clients and RentInfo entities are connected by “has rental logs for” because it give the rental log information/dates of clients. Cardinality is 1:N (one client can rent one or more properties so it can have more rental details in RentInfo table).

Properties and RentInfo entities are connected by “documents” because it documents when the properties are rented by the clients. Cardinality is 1:N (each property can have multiple rental information if it’s been rented multiple times).

Properties and Owners entities are connected by “belong to” because properties belong to a single owner. Cardinality is 1:N (one owner can have multiple properties but one property has only one owner).

Chen notation diagram with keys information:



3. Using DDL (Data Definition Language), create your database and tables on MySQL

MySQL text version of creating a database and tables:

```
#drop database if exists
drop database ClientRentalDB;

#create ClientRentalDB
CREATE DATABASE IF NOT EXISTS ClientRentalDB;

#show existing databases
SHOW databases;

#Use database
USE ClientRentalDB;

#create all the tables

#Clients table
DROP TABLE IF EXISTS Clients;
CREATE TABLE IF NOT EXISTS Clients(
   Client_No VARCHAR(20) NOT NULL PRIMARY KEY,         -- Unique client number with auto-increment,
declared as the primary key
   Client_Name VARCHAR(100) NOT NULL                -- Fullname of client, cannot be NULL
   );

DESCRIBE Clients;

#Owners table
DROP TABLE IF EXISTS Owners;
CREATE TABLE IF NOT EXISTS Owners(
   Owner_No VARCHAR(20) NOT NULL PRIMARY KEY,         -- Unique owner number with auto-increment,
declared as the primary key
   Owner_Name VARCHAR(100) NOT NULL                -- Fullname of owner, cannot be NULL
   );
DESCRIBE Owners;

#Properties table
DROP TABLE IF EXISTS Properties;
CREATE TABLE IF NOT EXISTS Properties(
            Property_No VARCHAR(20) PRIMARY KEY,       -- Unique property number with auto-increment,
declared as the primary key
   Property_Address VARCHAR(200) NOT NULL,          -- Address of property, constraint of value being
always provided
   Monthly_rent DECIMAL (10, 0) NOT NULL,           -- Rent amount monthly for up to 10 digits, but not
fractions
   Owner_No VARCHAR(20) NOT NULL,                      -- Calling foreign key from Owners table
   FOREIGN KEY (Owner_No) REFERENCES Owners (Owner_No) ON DELETE CASCADE    -- if owner is deleted
their property also will be deleted
   );

DESCRIBE Properties;
```

```
#RentInfo table
DROP TABLE IF EXISTS RentInfo;
CREATE TABLE IF NOT EXISTS RentInfo(
   Client_No VARCHAR(20) NOT NULL,                    -- Foreign key from Client table
   Property_No  VARCHAR(20) NOT NULL,                    -- Foreign key from Properties table
   Rent_start DATE NOT NULL,                    -- Starting date of renting the property
   Rent_finish DATE NOT NULL,                    -- Finishing date of renting the property
   PRIMARY KEY (Client_No, Property_No),           -- Composite key = Client_No + Property_No
   FOREIGN KEY (Client_No) REFERENCES Clients(Client_No) ON DELETE CASCADE,  -- If a client is deleted
their rental log will be deleted
   FOREIGN KEY (Property_No) REFERENCES Properties(Property_No) ON DELETE CASCADE -- If a property is
deleted, their rental log will be deleted
   );

DESCRIBE RentInfo;

show tables;
```

Action output after executing each row in creation of database and tables:

Output

Action Output

#	Time	Action	Message	Duration / Fetch
219	22:30:50	drop database ClientRentalDB	3 row(s) affected	0.031 sec
220	22:30:50	CREATE DATABASE IF NOT EXISTS ClientRentalDB	1 row(s) affected	0.016 sec
221	22:30:50	SHOW databases	10 row(s) returned	0.000 sec / 0.000 sec
222	22:30:50	USE ClientRentalDB	0 row(s) affected	0.000 sec
223	22:30:50	DROP TABLE IF EXISTS Clients	0 row(s) affected, 1 warning(s): 1051 Unknown table 'clientrentaldb.clients'	0.000 sec
224	22:30:50	CREATE TABLE IF NOT EXISTS Clients(Client_No VARCHAR(5) NOT NULL PRIMARY KEY, ...	0 row(s) affected	0.031 sec
225	22:30:50	DESCRIBE Clients	2 row(s) returned	0.000 sec / 0.000 sec
226	22:30:50	DROP TABLE IF EXISTS Owners	0 row(s) affected, 1 warning(s): 1051 Unknown table 'clientrentaldb.owners'	0.000 sec
227	22:30:50	CREATE TABLE IF NOT EXISTS Owners(Owner_No VARCHAR(5) NOT NULL PRIMARY KEY, ...	0 row(s) affected	0.031 sec
228	22:30:50	DESCRIBE Owners	2 row(s) returned	0.000 sec / 0.000 sec
229	22:30:50	DROP TABLE IF EXISTS Properties	0 row(s) affected, 1 warning(s): 1051 Unknown table 'clientrentaldb.properties'	0.000 sec
230	22:30:50	CREATE TABLE IF NOT EXISTS Properties(Property_No VARCHAR(5) PRIMARY KEY, -- Uniqu...	0 row(s) affected	0.031 sec
231	22:30:50	DESCRIBE Properties	4 row(s) returned	0.000 sec / 0.000 sec
232	22:30:50	DROP TABLE IF EXISTS RentInfo	0 row(s) affected, 1 warning(s): 1051 Unknown table 'clientrentaldb.rentinfo'	0.000 sec
233	22:30:50	CREATE TABLE IF NOT EXISTS RentInfo(Client_No VARCHAR(5) NOT NULL, -- ...	0 row(s) affected	0.031 sec
234	22:30:50	DESCRIBE RentInfo	4 row(s) returned	0.015 sec / 0.000 sec
235	22:30:50	show tables	4 row(s) returned	0.000 sec / 0.000 sec

Note: In the creation of database and tables, each explanation is included beside the commands. But if you look at the drop commands it's not necessary to include it but drop command is better for precaution if it's for testing the database and adding new features. That's why drop command has been used in this case.

Result in MySQL workbench:

MySQL command of creating ClientRental database:

```
#drop database if exists
drop database ClientRentalDB;

#create ClientRentalDB
CREATE DATABASE IF NOT EXISTS ClientRentalDB;

#show existing databases
SHOW databases;
```

MySQL result of the creation of ClientRental database:

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Database
clientrentaldb
firstdb
information_schema
library
mysql
performance_schema
sales3
school
school2
sys

MySQL command of creating Clients table:

```
#Use database
USE ClientRentalDB;

#create all the tables

#Clients table
DROP TABLE IF EXISTS Clients;
CREATE TABLE IF NOT EXISTS Clients(
    Client_No VARCHAR(20) NOT NULL PRIMARY KEY,        -- Unique client number with auto-increment, declared as the primary key
    Client_Name VARCHAR(100) NOT NULL                  -- Fullname of client, cannot be NULL
    );

DESCRIBE Clients;
```

MySQL result of the creation of Clients table:

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
Client_No	varchar(20)	NO	PRI	NULL	
Client_Name	varchar(100)	NO		NULL	

MySQL command of creating Owners table:

```
#Owners table
DROP TABLE IF EXISTS Owners;
CREATE TABLE IF NOT EXISTS Owners(
    Owner_No VARCHAR(20) NOT NULL PRIMARY KEY,         -- Unique owner number with auto-increment, declared as the primary key
    Owner_Name VARCHAR(100) NOT NULL                   -- Fullname of owner, cannot be NULL
    );
DESCRIBE Owners;
```

MySQL result of the creation of Owners table:

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
Owner_No	varchar(20)	NO	PRI	NULL	
Owner_Name	varchar(100)	NO		NULL	

MySQL command of creating Properties table:

```
#Properties table
DROP TABLE IF EXISTS Properties;
CREATE TABLE IF NOT EXISTS Properties(
    Property_No VARCHAR(20) PRIMARY KEY,        -- Unique property number with auto-increment, declared as the primary key
    Property_Address VARCHAR(200) NOT NULL,            -- Address of property, constraint of value being always provided
    Monthly_rent DECIMAL (10, 0) NOT NULL,             -- Rent amount monthly for up to 10 digits, but not fractions
    Owner_No VARCHAR(20) NOT NULL,                             -- Calling foreign key from Owners table
    FOREIGN KEY (Owner_No) REFERENCES Owners (Owner_No) ON DELETE CASCADE    -- if owner is deleted their property also will be deleted
    );

DESCRIBE Properties;
```

MySQL result of the creation of Properties table:

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
Property_No	varchar(20)	NO	PRI	NULL	
Property_Address	varchar(200)	NO		NULL	
Monthly_rent	decimal(10,0)	NO		NULL	
Owner_No	varchar(5)	NO	MUL	NULL	

MySQL command of creating RentInfo table:

```
#RentInfo table
DROP TABLE IF EXISTS RentInfo;
CREATE TABLE IF NOT EXISTS RentInfo(
    Client_No VARCHAR(20) NOT NULL,                        -- Foreign key from Client table
    Property_No  VARCHAR(20) NOT NULL,                      -- Foreign key from Properties table
    Rent_start DATE NOT NULL,                        -- Starting date of renting the property
    Rent_finish DATE NOT NULL,                       -- Finishing date of renting the property
    PRIMARY KEY (Client_No, Property_No),            -- Composite key = Client_No + Property_No
    FOREIGN KEY (Client_No) REFERENCES Clients(Client_No) ON DELETE CASCADE,  -- If a client is deleted their rental log will be deleted
    FOREIGN KEY (Property_No) REFERENCES Properties(Property_No) ON DELETE CASCADE -- If a property is deleted, their rental log will be deleted
    );

DESCRIBE RentInfo;
```

MySQL result of the creation of RentInfo table:

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
Client_No	varchar(20)	NO	PRI	NULL	
Property_No	varchar(20)	NO	PRI	NULL	
Rent_start	date	NO		NULL	
Rent_finish	date	NO		NULL	

MySQL command of showing all the tables created so far:

```
show tables;
```

MySQL result of all the tables created so far:

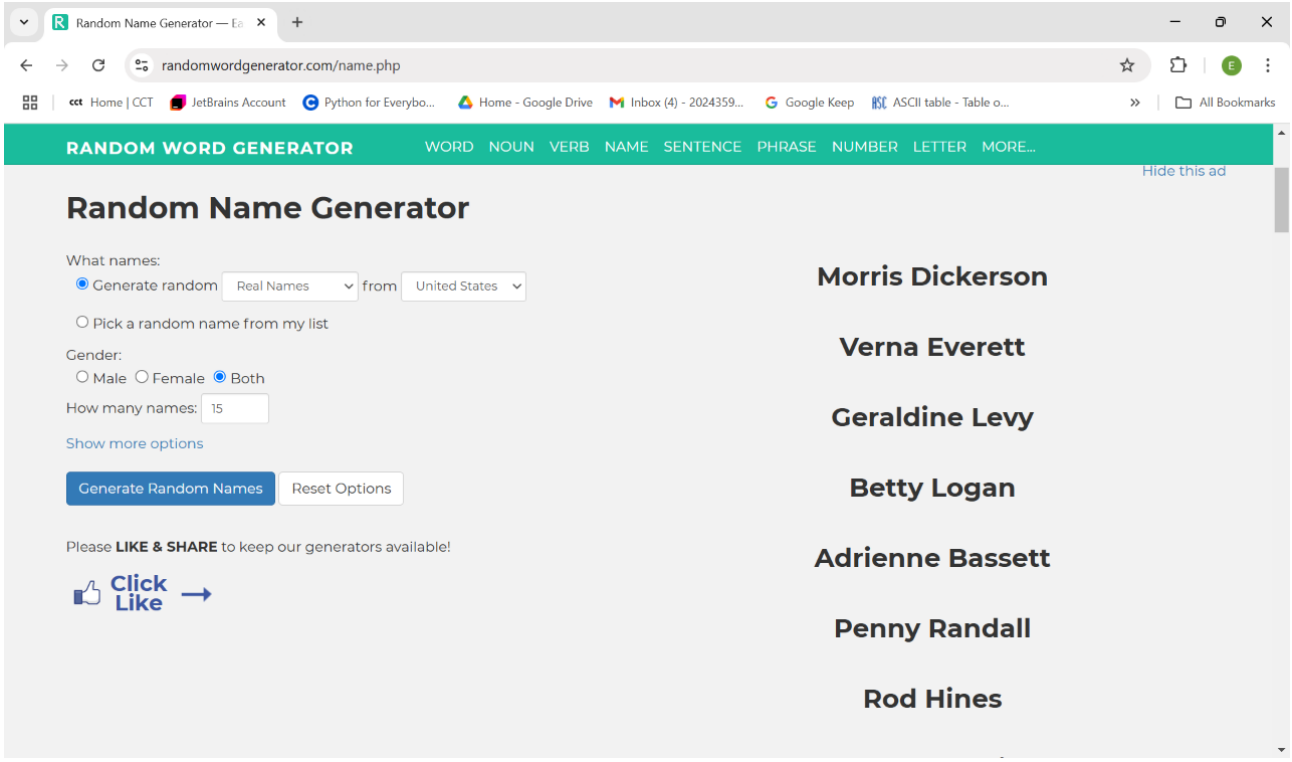
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Tables_in_clientrentaldb
clients
owners
properties
rentinfo

4.Insert at least the following information using dummy data and DML

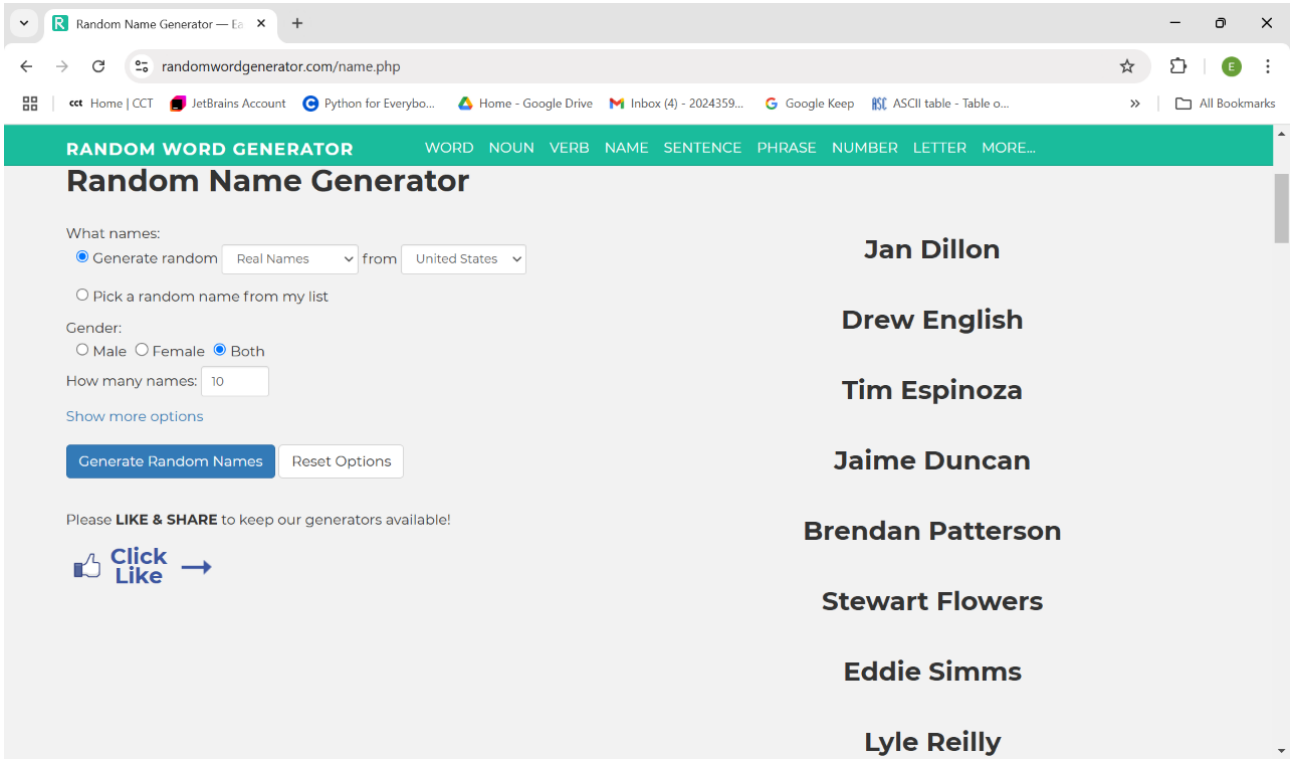
Note: Before inserting dummy data into the tables, Random Word Generator (Generator, 2024) has been used for both client and owner names.

Client names:



1. Morris Dickerson
2. Derna Everett
3. Geraldine Levy
4. Betty Logan
5. Adrienne Bassett
6. Denny Randall
7. Rod Hines
8. Yvette Austin
9. Donald Duncan
10. Russ Francis
11. Jeannette Dodge
12. Felicia Weston
13. Cesar Baker
14. Olivia Cline
15. Gabriel Whitaker

Owner names:



1. Jan Dillon
2. Drew English
3. Tim Espinoza
4. Jaime Duncan
5. Brendan Patterson
6. Stewart Flowers
7. Eddie Simms
8. Lyle Reilly
9. Marguerite Morrison
10. Roger Eldridge

MySQL text version of inserting dummy data into tables:

```
#Data Insertion

#Insert Data into Clients Table
INSERT INTO Clients (Client_No, Client_Name)        -- Inserting client names and numbers to Cliens table
VALUES                                              -- Every name and number are dummy datas
('CR-01', 'Morris Dickerson'),
('CR-02', 'Derna Everett'),
('CR-03', 'Geraldine Levy'),
('CR-04', 'Betty Logan'),
('CR-05', 'Adrienne Basset'),
('CR-06', 'Denny Randall'),
('CR-07', 'Rod Hines'),
('CR-08', 'Yvette Austin'),
('CR-09', 'Donald Duncan'),
('CR-10', 'Russ Francis'),
('CR-11', 'Jeanette Dodge'),
('CR-12', 'Felicia Weston'),
('CR-13', 'Cesar Baker'),
('CR-14', 'Olivia Cline'),
('CR-15', 'Gabriel Whitaker');

SELECT * FROM Clients;                              -- Shows the new Clients table with the inserted information

#Insert Data into Owners Table
INSERT INTO Owners (Owner_No, Owner_Name)           -- Inserting property owner names and numbers to
Owners table
VALUES                                              -- Every name and number are dummy datas
('C001', 'Jan Dillon'),
('C002', 'Drew English'),
('C003', 'Tim Espinoza'),
('C004', 'Jaime Duncan'),
('C005', 'Brendan Patterson'),
('C006', 'Stewart Flowers'),
('C007', 'Eddie Simms'),
('C008', 'Lyle Reilly'),
('C009', 'Marguerite Morrison'),
('C010', 'Roger Eldridge');

SELECT * FROM Owners;                               -- Shows the new Owners table with the inserted information

#Insert Data into Properties Table
INSERT INTO Properties (Property_No, Property_Address, Monthly_rent, Owner_No)    -- Inserting property
information to Properties table
VALUES                                              -- Every address and numbers are dummy datas
('PG001', '1 South Circular Road, Dublin', '1000','C001'),
('PG002', '2 South Circular Road, Dublin', '1100','C002'),
('PG003', '3 South Circular Road, Dublin', '1200','C003'),
('PG004', '4 South Circular Road, Dublin', '1300','C004'),
```

```
('PG005', '5 South Circular Road, Dublin', '1400','C005'),
('PG006', '6 South Circular Road, Dublin', '1500','C006'),
('PG007', '7 South Circular Road, Dublin', '1600','C007'),
('PG008', '8 South Circular Road, Dublin', '1700','C008'),
('PG009', '9 South Circular Road, Dublin', '1800','C009'),
('PG010', '10 South Circular Road, Dublin', '1900','C010'),
('PG011', '11 South Circular Road, Dublin', '2000','C001'),
('PG012', '12 South Circular Road, Dublin', '2100','C002'),
('PG013', '13 South Circular Road, Dublin', '2200','C003'),
('PG014', '14 South Circular Road, Dublin', '2300','C004'),
('PG015', '15 South Circular Road, Dublin', '2400','C005');

SELECT * FROM Properties;                    -- Shows the new Properties table with the
inserted information

#Insert Data into RentInfo Table
INSERT INTO RentInfo (Client_No, Property_No, Rent_start, Rent_finish)     -- Inserting rent information
to RentInfo table
VALUES                                  -- Every dates and numbers are dummy datas
('CR-01', 'PG001', '2021-06-01', '2021-08-01' ),
('CR-02', 'PG002', '2021-08-01', '2021-10-01' ),
('CR-03', 'PG003', '2021-10-01', '2021-12-01' ),
('CR-04', 'PG004', '2021-12-01', '2022-02-01' ),
('CR-05', 'PG005', '2022-02-01', '2022-04-01' ),
('CR-06', 'PG006', '2022-04-01', '2022-06-01' ),
('CR-07', 'PG007', '2022-06-01', '2022-08-01' ),
('CR-08', 'PG008', '2022-08-01', '2022-10-01' ),
('CR-09', 'PG009', '2022-10-01', '2022-12-01' ),
('CR-10', 'PG010', '2022-12-01', '2023-02-01' ),
('CR-11', 'PG011', '2023-02-01', '2023-04-01' ),
('CR-12', 'PG012', '2023-04-01', '2023-06-01' ),
('CR-13', 'PG013', '2023-06-01', '2023-08-01' ),
('CR-14', 'PG014', '2023-08-01', '2023-10-01' ),
('CR-15', 'PG015', '2023-10-01', '2024-01-01' );

SELECT * FROM RentInfo;                     -- Shows the new RentInfo table with the inserted
information
```

Action Output after executing each row in inserting dummy data into tables:

Output

Action Output

#	Time	Action	Message	Duration / Fetch
426	04:46:03	INSERT INTO Clients (Client_No, Client_Name) -- Inserting client names and numbers to Cliens tabl...	15 row(s) affected Records: 15 Duplicates: 0 Warnings: 0	0.000 sec
427	04:46:03	SELECT * FROM Clients LIMIT 0, 1000	15 row(s) returned	0.000 sec / 0.000 sec
428	04:46:03	INSERT INTO Owners (Owner_No, Owner_Name) -- Inserting property owner names and numbers...	10 row(s) affected Records: 10 Duplicates: 0 Warnings: 0	0.015 sec
429	04:46:03	SELECT * FROM Owners LIMIT 0, 1000	10 row(s) returned	0.000 sec / 0.000 sec
430	04:46:03	INSERT INTO Properties (Property_No, Property_Address, Monthly_rent, Owner_No) -- Inserting proper...	15 row(s) affected Records: 15 Duplicates: 0 Warnings: 0	0.000 sec
431	04:46:03	SELECT * FROM Properties LIMIT 0, 1000	15 row(s) returned	0.000 sec / 0.000 sec
432	04:46:03	INSERT INTO RentInfo (Client_No, Property_No, Rent_start, Rent_finish) -- Inserting rent informati...	15 row(s) affected Records: 15 Duplicates: 0 Warnings: 0	0.000 sec
433	04:46:03	SELECT * FROM RentInfo LIMIT 0, 1000	15 row(s) returned	0.000 sec / 0.000 sec

MySQL command of inserting data into Clients:

```
                    #Data Insertion

#Insert Data into Clients Table
INSERT INTO Clients (Client_No, Client_Name)          -- Inserting client names and numbers to Cliens table
VALUES                                                -- Every name and number are dummy datas
('CR-01', 'Morris Dickerson'),
('CR-02', 'Derna Everett'),
('CR-03', 'Geraldine Levy'),
('CR-04', 'Betty Logan'),
('CR-05', 'Adrienne Basset'),
('CR-06', 'Denny Randall'),
('CR-07', 'Rod Hines'),
('CR-08', 'Yvette Austin'),
('CR-09', 'Donald Duncan'),
('CR-10', 'Russ Francis'),
('CR-11', 'Jeanette Dodge'),
('CR-12', 'Felicia Weston'),
('CR-13', 'Cesar Baker'),
('CR-14', 'Olivia Cline'),
('CR-15', 'Gabriel Whitaker');

SELECT * FROM Clients;                          -- Shows the new Clients table with the inserted information
```

MySQL result of inserting data into Clients:

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Client_No	Client_Name
CR-01	Morris Dickerson
CR-02	Derna Everett
CR-03	Geraldine Levy
CR-04	Betty Logan
CR-05	Adrienne Basset
CR-06	Denny Randall
CR-07	Rod Hines
CR-08	Yvette Austin
CR-09	Donald Duncan
CR-10	Russ Francis
CR-11	Jeanette Dodge
CR-12	Felicia Weston
CR-13	Cesar Baker
CR-14	Olivia Cline
CR-15	Gabriel Whitaker

MySQL command of inserting data into Owners:

```
#Insert Data into Owners Table
INSERT INTO Owners (Owner_No, Owner_Name)          -- Inserting property owner names and numbers to Owners table
VALUES                                             -- Every name and number are dummy datas
('C001', 'Jan Dillon'),
('C002', 'Drew English'),
('C003', 'Tim Espinoza'),
('C004', 'Jaime Duncan'),
('C005', 'Brendan Patterson'),
('C006', 'Stewart Flowers'),
('C007', 'Eddie Simms'),
('C008', 'Lyle Reilly'),
('C009', 'Marguerite Morrison'),
('C010', 'Roger Eldridge');

SELECT * FROM Owners;                              -- Shows the new Owners table with the inserted information
```

MySQL result of inserting data into Owners:

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Owner_No	Owner_Name
C001	Jan Dillon
C002	Drew English
C003	Tim Espinoza
C004	Jaime Duncan
C005	Brendan Patterson
C006	Stewart Flowers
C007	Eddie Simms
C008	Lyle Reilly
C009	Marguerite Morrison
C010	Roger Eldridge

MySQL command of inserting data into Properties:

```
#Insert Data into Properties Table
INSERT INTO Properties (Property_No, Property_Address, Monthly_rent, Owner_No)    -- Inserting property information to Properties table
VALUES                                                                            -- Every address and numbers are dummy datas
('PG001', '1 South Circular Road, Dublin', '1000','C001'),
('PG002', '2 South Circular Road, Dublin', '1100','C002'),
('PG003', '3 South Circular Road, Dublin', '1200','C003'),
('PG004', '4 South Circular Road, Dublin', '1300','C004'),
('PG005', '5 South Circular Road, Dublin', '1400','C005'),
('PG006', '6 South Circular Road, Dublin', '1500','C006'),
('PG007', '7 South Circular Road, Dublin', '1600','C007'),
('PG008', '8 South Circular Road, Dublin', '1700','C008'),
('PG009', '9 South Circular Road, Dublin', '1800','C009'),
('PG010', '10 South Circular Road, Dublin', '1900','C010'),
('PG011', '11 South Circular Road, Dublin', '2000','C001'),
('PG012', '12 South Circular Road, Dublin', '2100','C002'),
('PG013', '13 South Circular Road, Dublin', '2200','C003'),
('PG014', '14 South Circular Road, Dublin', '2300','C004'),
('PG015', '15 South Circular Road, Dublin', '2400','C005');

SELECT * FROM Properties;                                                         -- Shows the new Properties table with the inserted information
```

MySQL result of inserting data into Owners:

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Property_No	Property_Address	Monthly_rent	Owner_No
PG001	1 South Circular Road, Dublin	1000	C001
PG002	2 South Circular Road, Dublin	1100	C002
PG003	3 South Circular Road, Dublin	1200	C003
PG004	4 South Circular Road, Dublin	1300	C004
PG005	5 South Circular Road, Dublin	1400	C005
PG006	6 South Circular Road, Dublin	1500	C006
PG007	7 South Circular Road, Dublin	1600	C007
PG008	8 South Circular Road, Dublin	1700	C008
PG009	9 South Circular Road, Dublin	1800	C009
PG010	10 South Circular Road, Dublin	1900	C010
PG011	11 South Circular Road, Dublin	2000	C001
PG012	12 South Circular Road, Dublin	2100	C002
PG013	13 South Circular Road, Dublin	2200	C003
PG014	14 South Circular Road, Dublin	2300	C004
PG015	15 South Circular Road, Dublin	2400	C005

MySQL command of inserting data into RentInfo:

```
#Insert Data into RentInfo Table
INSERT INTO RentInfo (Client_No, Property_No, Rent_start, Rent_finish)          -- Inserting rent information to RentInfo table
VALUES                                                                          -- Every dates and numbers are dummy datas
('CR-01', 'PG001', '2021-06-01', '2021-08-01' ),
('CR-02', 'PG002', '2021-08-01', '2021-10-01' ),
('CR-03', 'PG003', '2021-10-01', '2021-12-01' ),
('CR-04', 'PG004', '2021-12-01', '2022-02-01' ),
('CR-05', 'PG005', '2022-02-01', '2022-04-01' ),
('CR-06', 'PG006', '2022-04-01', '2022-06-01' ),
('CR-07', 'PG007', '2022-06-01', '2022-08-01' ),
('CR-08', 'PG008', '2022-08-01', '2022-10-01' ),
('CR-09', 'PG009', '2022-10-01', '2022-12-01' ),
('CR-10', 'PG010', '2022-12-01', '2023-02-01' ),
('CR-11', 'PG011', '2023-02-01', '2023-04-01' ),
('CR-12', 'PG012', '2023-04-01', '2023-06-01' ),
('CR-13', 'PG013', '2023-06-01', '2023-08-01' ),
('CR-14', 'PG014', '2023-08-01', '2023-10-01' ),
('CR-15', 'PG015', '2023-10-01', '2024-01-01' );

SELECT * FROM Clients;                                                          -- Shows the new RentInfo table with the inserted information
```

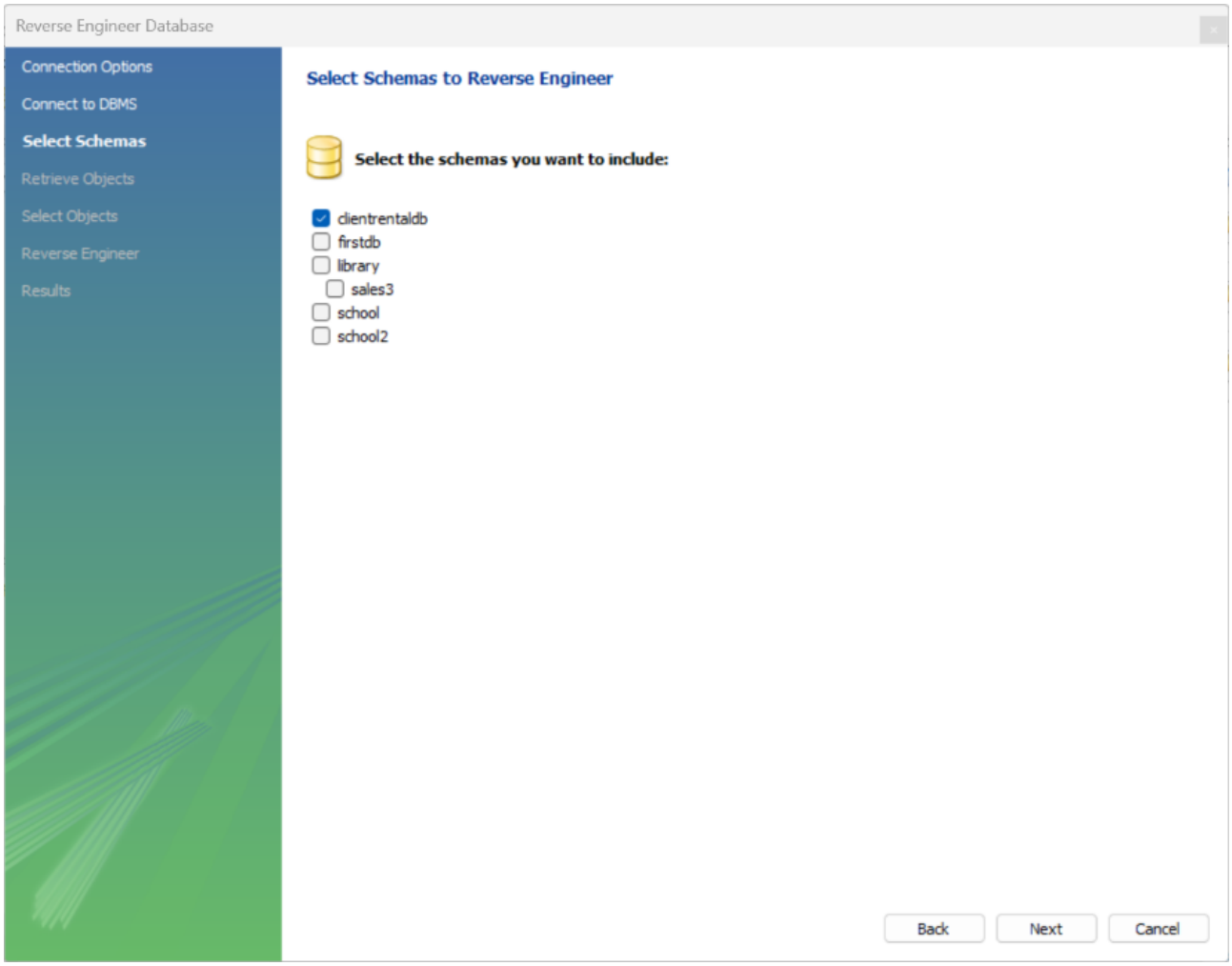
MySQL result of inserting data into RentInfo:

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

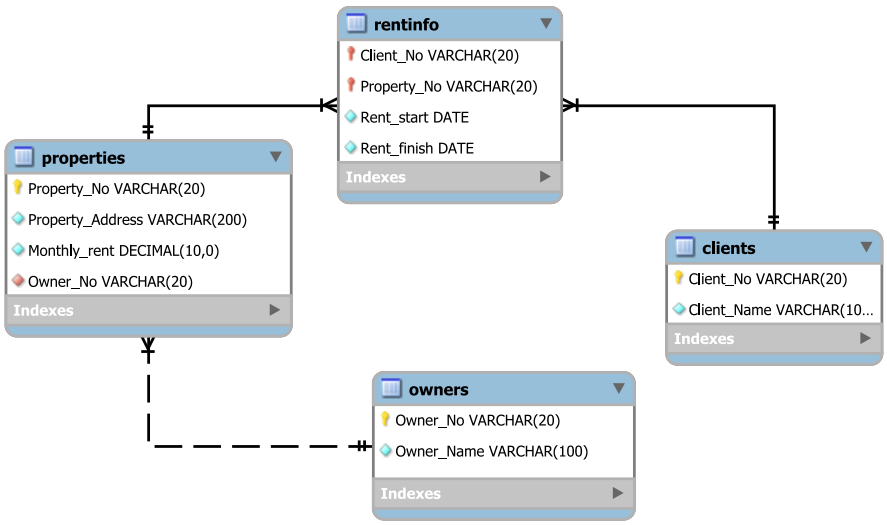
Client_No	Property_No	Rent_start	Rent_finish
CR-01	PG001	2021-06-01	2021-08-01
CR-02	PG002	2021-08-01	2021-10-01
CR-03	PG003	2021-10-01	2021-12-01
CR-04	PG004	2021-12-01	2022-02-01
CR-05	PG005	2022-02-01	2022-04-01
CR-06	PG006	2022-04-01	2022-06-01
CR-07	PG007	2022-06-01	2022-08-01
CR-08	PG008	2022-08-01	2022-10-01
CR-09	PG009	2022-10-01	2022-12-01
CR-10	PG010	2022-12-01	2023-02-01
CR-11	PG011	2023-02-01	2023-04-01
CR-12	PG012	2023-04-01	2023-06-01
CR-13	PG013	2023-06-01	2023-08-01
CR-14	PG014	2023-08-01	2023-10-01
CR-15	PG015	2023-10-01	2024-01-01

5. Apply reverse engineering to the created tables and produce the ERD using Crow's Foot notation

Note: After creating all the tables using DDL and inserting dummy data into the tables using DML, in MySQL workbench, ERD by Crow's Foot notation has been made. Database->Reverse Engineer



The Result of ERD Diagram using Crow's Foot notation



1.2 Database Part 2

MySQL text version of creating queries in ClientRentalDB:

#Queries

#1. Retrieve all clients along with their associated properties.

-- Option 1: Gives all the information including Client No, Name and Property No, Address and monthly rent amount

```
SELECT
                    -- Joining Clients, Properties and RentInfo tables to show clients and their rented properties

   Clients.Client_No AS Client_No,                                                    -- Showing Client number from Clients table

   Clients.Client_Name AS Client_Name,                                                -- Showing Client name from Clients table

   Properties.Property_No AS Property_No,                                             -- Showing Property number from Properties table

   Properties.Property_Address AS Property_Address,                   -- Showing Property Address from Properties table

   Properties.Monthly_rent AS Monthly_rent                                            -- Showting monthly rent amount from Properties table

FROM

   RentInfo

JOIN
                    --  Linking RentInfo table to Clients table by Client_No in both tables

   Clients ON RentInfo.Client_No = Clients.Client_No

JOIN
                    -- Linking RentInfo table to Properties table by Property_No in both tables

   Properties ON RentInfo.Property_No = Properties.Property_No;
```

-- Option 2: Only the client name and property address

```
SELECT
                    -- Joining Clients, Properties and RentInfo tables to show clients and their rented properties

         Clients.Client_Name AS Client_Name,                                          -- Showing Client Names from Clients table

         Properties.Property_Address AS Property_Address                   -- Showing Property Address from Properties table

FROM

   RentInfo

JOIN
                    --  Linking RentInfo table to Clients table by Client_No in both tables
```

```
   Clients ON RentInfo.Client_No = Clients.Client_No

JOIN
                    -- Linking RentInfo table to Properties table by Property_No in both tables

   Properties ON RentInfo.Property_No = Properties.Property_No;
```

#2. List all properties rented out by all clients whose name begins with ‘D’.

```
-- Option 1: Giving client names with letter 'D' and the property number and address information

SELECT
                    -- Joining Clients, Properties and RentInfo tables to show clients that their names start with
letter 'D', and their rented properties

          Clients.Client_Name AS Client_Name_D,                                                    -- Showing
Client Names  from Clients table

   Properties.Property_No AS Property_No,                                          -- Showing Property
number from Properties table

   Properties.Property_Address AS Property_Address                        -- Showing Property Address from
Properties table

FROM

          RentInfo

JOIN
                    --  Linking RentInfo table to Clients table by Client_No in both tables

          Clients ON RentInfo.Client_No = Clients.Client_No

JOIN
                    -- Linking RentInfo table to Properties table by Property_No in both tables

          Properties ON RentInfo.Property_No = Properties.Property_No

WHERE Clients.Client_Name LIKE 'D%';                                                                --
Choosing client names that starts with letter 'D' using LIKE operator to match the values

-- Option 2: Only showing the properties from the clients whose name starts with letter 'D'

SELECT
                    -- Joining Clients, Properties and RentInfo tables to show clients that their names start with
letter 'D', and their rented properties

   Properties.Property_No AS Property_No,                                          -- Showing Property
number from Properties table

   Properties.Property_Address AS Property_Address                        -- Showing Property Address from
Properties table

FROM

          RentInfo

JOIN
                    --  Linking RentInfo table to Clients table by Client_No in both tables

          Clients ON RentInfo.Client_No = Clients.Client_No
```

```
JOIN
                    -- Linking RentInfo table to Properties table by Property_No in both tables

        Properties ON RentInfo.Property_No = Properties.Property_No

WHERE Clients.Client_Name LIKE 'D%';                                                    --
Choosing client names that starts with letter 'D' using LIKE operator to match the values
```

#3. List all clients who have properties rented out for a specific duration, from the date 2023-02-20 to 2023-10-20.

```
-- Option 1: Listing all clients with specific date range with their rent date information

SELECT
                    -- Joining Clients and RentInfo table to list all the client who rented properties between
2023-02-20 to 2023-10-20

        Clients.Client_No,
        -- Showing Client number from Clients table

   Clients.Client_Name,                                                                 -- Showing
Client Names  from Clients table

   RentInfo.Rent_start,                                                                 -- Showing
start date  from RentInfo table

   RentInfo.Rent_finish                                                                 -- Showing
finish date  from RentInfo table

FROM

        RentInfo

JOIN
                    --  Linking RentInfo table to Clients table by Client_No in both tables

        Clients ON RentInfo.Client_No = Clients.Client_No

WHERE
                    --  Finding clients who rented properties within specific date range

        RentInfo.Rent_start >= '2023-02-20'                                             -- if the
start date is after 2023-02-20

   AND

   RentInfo.Rent_finish <= '2023-10-20';                                        -- if the finish date is
before 2023-10-20

-- Option 2: Listing only the clients with specific date ranges without the rent date information

SELECT
                    -- Joining Clients and RentInfo table to list all the client who rented properties between
2023-02-20 to 2023-10-20

        Clients.Client_No,
        -- Showing Client number from Clients table

   Clients.Client_Name                                                                  -- Showing
Client Names  from Clients table
```

```
FROM
        RentInfo
JOIN
                -- Linking RentInfo table to Clients table by Client_No in both tables
        Clients ON RentInfo.Client_No = Clients.Client_No
WHERE
                -- Finding clients who rented properties within specific date range
        RentInfo.Rent_start >= '2023-02-20'                                    -- if the start date is after 2023-02-20
  AND
  RentInfo.Rent_finish <= '2023-10-20';                                        -- if the finish date is before 2023-10-20
```

#4. Calculate the total monthly rent for each client.

```
SELECT
                -- Joining Clients, Properties, RentInfo table to list all the client with their total monthly rent
        Clients.Client_No,
        -- Showing Client number from Clients table
  Clients.Client_Name,                                                         -- Showing Client Names  from Clients table
  SUM(Properties.Monthly_rent) AS Total_Monthly_rent                           -- Calculating the total monthly rent for each client using SUM
FROM
        RentInfo
JOIN
                -- Linking RentInfo table to Clients table by Client_No in both tables
        Clients ON RentInfo.Client_No = Clients.Client_No
JOIN
                -- Linking RentInfo table to Properties table by Property_No in both tables
        Properties ON RentInfo.Property_No = Properties.Property_No
GROUP BY
                -- Grouping by client to find the total rent amount
        Clients.Client_No,
  Clients.Client_Name;
```

#5. Find the owner of a specific property.

```
SELECT
                -- Joining Properties and Owners table to find the owner of a specific property
  Properties.Property_No,                                                      -- Showing Property number from Properties table
```

```
   Properties.Property_Address,                                                    -- Showing
Property Address from Properties table

   Owners.Owner_name
         -- Showing Owner name from Owners table

FROM

         Properties

JOIN
                  -- Linking Properties table to Owners table by Owner_No in both tables

         Owners ON Properties.Owner_No = Owners.Owner_No

WHERE
                  -- Find the owner of the property number 15

         Properties.Property_No = 'PG015';
```

#6. Count the total number of properties owned by each owner.

```
SELECT
                  -- Joining Properties and Owners table to find the total number of properties of each owner

   Owners.Owner_No,                                                                --
Showing Owner number from Owners table

   Owners.Owner_Name,                                                              --
Showing Owner name from Owners table

   COUNT(Properties.Property_No) AS Total_Properties_Count   -- Calculating property count from
Property table

FROM

   Owners

JOIN
                  -- Linking Owners table to Properties table to count properties by each owner

   Properties ON Owners.Owner_No = Properties.Owner_No

GROUP BY
                  -- Grouping by owner to calculate property count

   Owners.Owner_No,

   Owners.Owner_Name;
```

#7. Identify owners who own multiple properties.

```
SELECT
                  -- Joining Properties and Owners table to find the owners who owns multiple properties

   Owners.Owner_No,                                                                --
Showing Owner number from Owners table

   Owners.Owner_Name,                                                              --
Showing Owner name from Owners table
```

```
   COUNT(Properties.Property_No) AS Total_Properties_Count    -- Calculating property count from
Property table

FROM

   Owners

JOIN
                    -- Linking Owners table to Properties table to count properties by each owner

   Properties ON Owners.Owner_No = Properties.Owner_No

GROUP BY
                    -- Grouping by owner to calculate property count

   Owners.Owner_No,

   Owners.Owner_Name

HAVING
                              -- Finding the owners who have more than 1 property

         COUNT(Properties.Property_No) > 1;
```

#8. List all clients along with the total rent they pay annually, sorted in ascending order (i.e. lowest rent at the top).

```
SELECT
                    -- Joining Clients, Properties, RentInfo table to list all the client with their annual rent
sorted by ASC

         Clients.Client_No,
         -- Showing Client number from Clients table

   Clients.Client_Name,                                                                          -- Showing
Client Names  from Clients table

   SUM(Properties.Monthly_rent * 12) AS Annual_rent                  -- Calculating the total annual rent by
multiplying each amount with 12

FROM

         RentInfo

JOIN
                    --  Linking RentInfo table to Clients table by Client_No in both tables

         Clients ON RentInfo.Client_No = Clients.Client_No

JOIN
                    -- Linking RentInfo table to Properties table by Property_No in both tables

         Properties ON RentInfo.Property_No = Properties.Property_No

GROUP BY
                    -- Grouping by client to find the total annual rent amount

         Clients.Client_No,

   Clients.Client_Name
```

```
ORDER BY
                -- Sorting the annual rent by lowest to highest
        Annual_rent ASC;
```

#9. Find the client who pays the highest monthly rent.

```
SELECT
                -- Joining Clients, Properties, RentInfo table to list all the client with the highest monthly
rent
        Clients.Client_No,
        -- Showing Client number from Clients table
  Clients.Client_Name,                                                                    -- Showing
Client Names  from Clients table
  Properties.Monthly_rent AS Highest_rent                                  -- Showing the highest
amount of rent
FROM
        RentInfo
JOIN
                -- Linking RentInfo table to Clients table by Client_No in both tables
        Clients ON RentInfo.Client_No = Clients.Client_No
JOIN
                -- Linking RentInfo table to Properties table by Property_No in both tables
        Properties ON RentInfo.Property_No = Properties.Property_No
WHERE
                -- Finding the highest paying rent from properties using MAX by filtering Monthly_rent
        Properties.Monthly_Rent = (
                SELECT MAX(Monthly_rent)
    FROM Properties);
```

#10. List all properties with rent amounts greater than the average rent amount across all properties.

```
SELECT
                -- Finding the average monthly rent
        AVG(Monthly_rent)
FROM
        Properties;


SELECT
                -- Selecting properties with rents that has greater amount than average
  Properties.Property_No,                                                                 -- Showing
Property number from Properties table
```

```
    Properties.Property_Address,                                    -- Showing Property Address from Properties table

    Properties.Monthly_rent                                         -- Showing Monthly_rent from Properties table

FROM

        Properties

WHERE
            -- Find the property information that has more than average monthly rent

        Properties.Monthly_rent > (

            SELECT

                AVG(Monthly_rent)

        FROM

                Properties);
```

Action Output after executing each row in creating queries:

Output

Action Output

#	Time	Action	Message	Duration / Fetch
434	04:46:03	SELECT -- Joining Clients, Properties and RentInfo tables to show clients and their rented properties Cli...	15 row(s) returned	0.000 sec / 0.000 sec
435	04:46:03	SELECT -- Joining Clients, Properties and RentInfo tables to show clients and their rented properties Clien...	15 row(s) returned	0.000 sec / 0.000 sec
436	04:46:03	SELECT -- Joining Clients, Properties and RentInfo tables to show clients that their names start with letter ...	3 row(s) returned	0.000 sec / 0.000 sec
437	04:46:03	SELECT -- Joining Clients, Properties and RentInfo tables to show clients that their names start with letter ...	3 row(s) returned	0.000 sec / 0.000 sec
438	04:46:03	SELECT -- Joining Clients and RentInfo table to list all the client who rented properties between 2023-02-...	3 row(s) returned	0.000 sec / 0.000 sec
439	04:46:03	SELECT -- Joining Clients and RentInfo table to list all the client who rented properties between 2023-02-...	3 row(s) returned	0.000 sec / 0.000 sec
440	04:46:03	SELECT -- Joining Clients, Properties, RentInfo table to list all the client with their total monthly rent Clients...	15 row(s) returned	0.000 sec / 0.000 sec
441	04:46:03	SELECT -- Joining Properties and Owners table to find the owner of a specific property Properties.Pro...	1 row(s) returned	0.000 sec / 0.000 sec
442	04:46:03	SELECT -- Joining Properties and Owners table to find the total number of properties of each owner O...	10 row(s) returned	0.000 sec / 0.000 sec
443	04:46:03	SELECT -- Joining Properties and Owners table to find the owners who owns multiple properties Owne...	5 row(s) returned	0.000 sec / 0.000 sec
444	04:46:03	SELECT -- Joining Clients, Properties, RentInfo table to list all the client with their annual rent sorted by A...	15 row(s) returned	0.000 sec / 0.000 sec
445	04:46:03	SELECT -- Joining Clients, Properties, RentInfo table to list all the client with the highest monthly rent Clie...	1 row(s) returned	0.000 sec / 0.000 sec
446	04:46:03	SELECT-- Finding the average monthly rent AVG(Monthly_rent) FROM Properties LIMIT 0, 1000	1 row(s) returned	0.000 sec / 0.000 sec
447	04:46:03	SELECT -- Selecting properties with rents that has greater amount than average Properties.Property_N...	7 row(s) returned	0.000 sec / 0.000 sec

1. Retrieve all clients along with their associated properties.

Query 1 – Option 1

```
                #Queries
#1. Retrieve all clients along with their associated properties.
-- Option 1: Gives all the information including Client No, Name and Property No, Address and monthly rent amount

SELECT                                                  -- Joining Clients, Properties and RentInfo tables to show clients and their rented properties
    Clients.Client_No AS Client_No,                     -- Showing Client number from Clients table
    Clients.Client_Name AS Client_Name,                 -- Showing Client name from Clients table
    Properties.Property_No AS Property_No,              -- Showing Property number from Properties table
    Properties.Property_Address AS Property_Address,    -- Showing Property Address from Properties table
    Properties.Monthly_rent AS Monthly_rent             -- Showting monthly rent amount from Properties table
FROM
    RentInfo
JOIN                                                    --  Linking RentInfo table to Clients table by Client_No in both tables
    Clients ON RentInfo.Client_No = Clients.Client_No
JOIN                                                    -- Linking RentInfo table to Properties table by Property_No in both tables
    Properties ON RentInfo.Property_No = Properties.Property_No;
```

Result 1.1

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Client_No	Client_Name	Property_No	Property_Address	Monthly_rent
CR-01	Morris Dickerson	PG001	1 South Circular Road, Dublin	1000
CR-02	Derna Everett	PG002	2 South Circular Road, Dublin	1100
CR-03	Geraldine Levy	PG003	3 South Circular Road, Dublin	1200
CR-04	Betty Logan	PG004	4 South Circular Road, Dublin	1300
CR-05	Adrienne Basset	PG005	5 South Circular Road, Dublin	1400
CR-06	Denny Randall	PG006	6 South Circular Road, Dublin	1500
CR-07	Rod Hines	PG007	7 South Circular Road, Dublin	1600
CR-08	Yvette Austin	PG008	8 South Circular Road, Dublin	1700
CR-09	Donald Duncan	PG009	9 South Circular Road, Dublin	1800
CR-10	Russ Francis	PG010	10 South Circular Road, Dublin	1900
CR-11	Jeanette Dodge	PG011	11 South Circular Road, Dublin	2000
CR-12	Felicia Weston	PG012	12 South Circular Road, Dublin	2100
CR-13	Cesar Baker	PG013	13 South Circular Road, Dublin	2200
CR-14	Olivia Cline	PG014	14 South Circular Road, Dublin	2300
CR-15	Gabriel Whitaker	PG015	15 South Circular Road, Dublin	2400

Query 1 - Option 2

```sql
-- Option 2: Only the client name and property address
SELECT                                                  -- Joining Clients, Properties and RentInfo tables to show clients and their rented properties
    Clients.Client_Name AS Client_Name,                 -- Showing Client Names from Clients table
    Properties.Property_Address AS Property_Address     -- Showing Property Address from Properties table
FROM
    RentInfo
JOIN                                                    --  Linking RentInfo table to Clients table by Client_No in both tables
    Clients ON RentInfo.Client_No = Clients.Client_No
JOIN                                                    -- Linking RentInfo table to Properties table by Property_No in both tables
    Properties ON RentInfo.Property_No = Properties.Property_No;
```

Result 1.2

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Client_Name	Property_Address
Morris Dickerson	1 South Circular Road, Dublin
Derna Everett	2 South Circular Road, Dublin
Geraldine Levy	3 South Circular Road, Dublin
Betty Logan	4 South Circular Road, Dublin
Adrienne Basset	5 South Circular Road, Dublin
Denny Randall	6 South Circular Road, Dublin
Rod Hines	7 South Circular Road, Dublin
Yvette Austin	8 South Circular Road, Dublin
Donald Duncan	9 South Circular Road, Dublin
Russ Francis	10 South Circular Road, Dublin
Jeanette Dodge	11 South Circular Road, Dublin
Felicia Weston	12 South Circular Road, Dublin
Cesar Baker	13 South Circular Road, Dublin
Olivia Cline	14 South Circular Road, Dublin
Gabriel Whitaker	15 South Circular Road, Dublin

2. List all properties rented out by all clients whose name begins with ‘D’.

Query 2 – Option 1

```sql
#2. List all properties rented out by all clients whose name begins with ‘D’.
-- Option 1: Giving client names with letter 'D' and the property number and address information
SELECT                                                  -- Joining Clients, Properties and RentInfo tables to show clients that their names start with letter
    Clients.Client_Name AS Client_Name_D,               -- Showing Client Names  from Clients table
    Properties.Property_No AS Property_No,              -- Showing Property number from Properties table
    Properties.Property_Address AS Property_Address     -- Showing Property Address from Properties table
FROM
    RentInfo
JOIN                                                    --  Linking RentInfo table to Clients table by Client_No in both tables
    Clients ON RentInfo.Client_No = Clients.Client_No
JOIN                                                    -- Linking RentInfo table to Properties table by Property_No in both tables
    Properties ON RentInfo.Property_No = Properties.Property_No
WHERE Clients.Client_Name LIKE 'D%';                    -- Choosing client names that starts with letter 'D' using LIKE operator to match the values
```

Result 2.1

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Client_Name_D	Property_No	Property_Address
Derna Everett	PG002	2 South Circular Road, Dublin
Denny Randall	PG006	6 South Circular Road, Dublin
Donald Duncan	PG009	9 South Circular Road, Dublin

Query 2 - Option 2

```sql
-- Option 2: Only showing the properties from the clients whose name starts with letter 'D'
SELECT                                                  -- Joining Clients, Properties and RentInfo tables to show clients that their names start with letter
    Properties.Property_No AS Property_No,              -- Showing Property number from Properties table
    Properties.Property_Address AS Property_Address     -- Showing Property Address from Properties table
FROM
    RentInfo
JOIN                                                    --  Linking RentInfo table to Clients table by Client_No in both tables
    Clients ON RentInfo.Client_No = Clients.Client_No
JOIN                                                    -- Linking RentInfo table to Properties table by Property_No in both tables
    Properties ON RentInfo.Property_No = Properties.Property_No
WHERE Clients.Client_Name LIKE 'D%';                    -- Choosing client names that starts with letter 'D' using LIKE operator to match the values
```

Result 2.2

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Property_No	Property_Address
PG002	2 South Circular Road, Dublin
PG006	6 South Circular Road, Dublin
PG009	9 South Circular Road, Dublin

3. List all clients who have properties rented out for a specific duration, from the date 2023-02-20 to 2023- 10-20.

Query 3 – Option 1

```sql
#3. List all clients who have properties rented out for a specific duration, from the date 2023-02-20 to 2023-10-20.
-- Option 1: Listing all clients with specific date range with their rent date information
SELECT                                                  -- Joining Clients and RentInfo table to list all the client who rented properties between 2023-02-20
    Clients.Client_No,                                  -- Showing Client number from Clients table
    Clients.Client_Name,                                -- Showing Client Names  from Clients table
    RentInfo.Rent_start,                                -- Showing start date  from RentInfo table
    RentInfo.Rent_finish                                -- Showing finish date  from RentInfo table
FROM
    RentInfo
JOIN                                                    --  Linking RentInfo table to Clients table by Client_No in both tables
    Clients ON RentInfo.Client_No = Clients.Client_No
WHERE                                                   --  Finding clients who rented properties within specific date range
    RentInfo.Rent_start >= '2023-02-20'                 -- if the start date is after 2023-02-20
    AND
    RentInfo.Rent_finish <= '2023-10-20';               -- if the finish date is before 2023-10-20
```

Result 3.1

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Client_No	Client_Name	Rent_start	Rent_finish
CR-12	Felicia Weston	2023-04-01	2023-06-01
CR-13	Cesar Baker	2023-06-01	2023-08-01
CR-14	Olivia Cline	2023-08-01	2023-10-01

Query 3 - Option 2

```sql
-- Option 2: Listing only the clients with specific date ranges without the rent date information
SELECT                                                  -- Joining Clients and RentInfo table to list all the client who rented properties between 2023-02-20
    Clients.Client_No,                                  -- Showing Client number from Clients table
    Clients.Client_Name                             -- Showing Client Names  from Clients table
FROM
    RentInfo
JOIN                                                    --  Linking RentInfo table to Clients table by Client_No in both tables
    Clients ON RentInfo.Client_No = Clients.Client_No
WHERE                                                   --  Finding clients who rented properties within specific date range
    RentInfo.Rent_start >= '2023-02-20'                 -- if the start date is after 2023-02-20
    AND
    RentInfo.Rent_finish <= '2023-10-20';               -- if the finish date is before 2023-10-20
```

Result 3.2

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Client_No	Client_Name
CR-12	Felicia Weston
CR-13	Cesar Baker
CR-14	Olivia Cline

4. Calculate the total monthly rent for each client.

Query 4

```
#4. Calculate the total monthly rent for each client.
SELECT                                                  -- Joining Clients, Properties, RentInfo table to list all the client with their total monthly rent
    Clients.Client_No,                                  -- Showing Client number from Clients table
    Clients.Client_Name,                                -- Showing Client Names  from Clients table
    SUM(Properties.Monthly_rent) AS Total_Monthly_rent  -- Calculating the total monthly rent for each client using SUM
FROM
    RentInfo
JOIN                                                    --  Linking RentInfo table to Clients table by Client_No in both tables
    Clients ON RentInfo.Client_No = Clients.Client_No
JOIN                                                    -- Linking RentInfo table to Properties table by Property_No in both tables
    Properties ON RentInfo.Property_No = Properties.Property_No
GROUP BY                                                -- Grouping by client to find the total rent amount
    Clients.Client_No,
    Clients.Client_Name;
```

Result 4

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Client_No	Client_Name	Total_Monthly_rent
CR-01	Morris Dickerson	1000
CR-02	Derna Everett	1100
CR-03	Geraldine Levy	1200
CR-04	Betty Logan	1300
CR-05	Adrienne Basset	1400
CR-06	Denny Randall	1500
CR-07	Rod Hines	1600
CR-08	Yvette Austin	1700
CR-09	Donald Duncan	1800
CR-10	Russ Francis	1900
CR-11	Jeanette Dodge	2000
CR-12	Felicia Weston	2100
CR-13	Cesar Baker	2200
CR-14	Olivia Cline	2300
CR-15	Gabriel Whitaker	2400

5. Find the owner of a specific property.

Query 5

```
#5. Find the owner of a specific property.
SELECT                                          -- Joining Properties and Owners table to find the owner of a specific property
    Properties.Property_No,                     -- Showing Property number from Properties table
    Properties.Property_Address,                -- Showing Property Address from Properties table
    Owners.Owner_name                           -- Showing Owner name from Owners table
FROM
    Properties
JOIN                                            -- Linking Properties table to Owners table by Owner_No in both tables
    Owners ON Properties.Owner_No = Owners.Owner_No
WHERE                                           -- Find the owner of the property number 15
    Properties.Property_No = 'PG015';
```

Result 5

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Property_No	Property_Address	Owner_name
PG015	15 South Circular Road, Dublin	Brendan Patterson

6. Count the total number of properties owned by each owner.

Query 6

```sql
#6. Count the total number of properties owned by each owner.
SELECT                                                  -- Joining Properties and Owners table to find the total number of properties of each owner
    Owners.Owner_No,                                    -- Showing Owner number from Owners table
    Owners.Owner_Name,                                  -- Showing Owner name from Owners table
    COUNT(Properties.Property_No) AS Total_Properties_Count   -- Calculating property count from Property table
FROM
    Owners
JOIN                                                    -- Linking Owners table to Properties table to count properties by each owner
    Properties ON Owners.Owner_No = Properties.Owner_No
GROUP BY                                                -- Grouping by owner to calculate property count
    Owners.Owner_No,
    Owners.Owner_Name;
```

Result 6

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Owner_No	Owner_Name	Total_Properties_Count
C001	Jan Dillon	2
C002	Drew English	2
C003	Tim Espinoza	2
C004	Jaime Duncan	2
C005	Brendan Patterson	2
C006	Stewart Flowers	1
C007	Eddie Simms	1
C008	Lyle Reilly	1
C009	Marguerite Morrison	1
C010	Roger Eldridge	1

7. Identify owners who own multiple properties.

Query 7

```sql
#7. Identify owners who own multiple properties.
SELECT                                                  -- Joining Properties and Owners table to find the owners who owns multiple properties
    Owners.Owner_No,                                    -- Showing Owner number from Owners table
    Owners.Owner_Name,                                  -- Showing Owner name from Owners table
    COUNT(Properties.Property_No) AS Total_Properties_Count   -- Calculating property count from Property table
FROM
    Owners
JOIN                                                    -- Linking Owners table to Properties table to count properties by each owner
    Properties ON Owners.Owner_No = Properties.Owner_No
GROUP BY                                                -- Grouping by owner to calculate property count
    Owners.Owner_No,
    Owners.Owner_Name
HAVING                                                  -- Finding the owners who have more than 1 property
    COUNT(Properties.Property_No) > 1;
```

Result 7

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Owner_No	Owner_Name	Total_Properties_Count
C001	Jan Dillon	2
C002	Drew English	2
C003	Tim Espinoza	2
C004	Jaime Duncan	2
C005	Brendan Patterson	2

8. List all clients along with the total rent they pay annually, sorted in ascending order (i.e. lowest rent at the top).

Query 8

```sql
#8. List all clients along with the total rent they pay annually, sorted in ascending order (i.e. lowest rent at the top).
SELECT                                                  -- Joining Clients, Properties, RentInfo table to list all the client with their annual rent sorted by
    Clients.Client_No,                                  -- Showing Client number from Clients table
    Clients.Client_Name,                                -- Showing Client Names  from Clients table
    SUM(Properties.Monthly_rent * 12) AS Annual_rent    -- Calculating the total annual rent by multiplying each amount with 12
FROM
    RentInfo
JOIN                                                    --  Linking RentInfo table to Clients table by Client_No in both tables
    Clients ON RentInfo.Client_No = Clients.Client_No
JOIN                                                    -- Linking RentInfo table to Properties table by Property_No in both tables
    Properties ON RentInfo.Property_No = Properties.Property_No
GROUP BY                                                -- Grouping by client to find the total annual rent amount
    Clients.Client_No,
    Clients.Client_Name
ORDER BY                                                -- Sorting the annual rent by lowest to highest
    Annual_rent ASC;
```

Result 8

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Client_No	Client_Name	Annual_rent
CR-01	Morris Dickerson	12000
CR-02	Derna Everett	13200
CR-03	Geraldine Levy	14400
CR-04	Betty Logan	15600
CR-05	Adrienne Basset	16800
CR-06	Denny Randall	18000
CR-07	Rod Hines	19200
CR-08	Yvette Austin	20400
CR-09	Donald Duncan	21600
CR-10	Russ Francis	22800
CR-11	Jeanette Dodge	24000
CR-12	Felicia Weston	25200
CR-13	Cesar Baker	26400
CR-14	Olivia Cline	27600
CR-15	Gabriel Whitaker	28800

9. Find the client who pays the highest monthly rent.

Query 9

```sql
#9. Find the client who pays the highest monthly rent.
SELECT                                                  -- Joining Clients, Properties, RentInfo table to list all the client with the highest monthly rent
    Clients.Client_No,                                  -- Showing Client number from Clients table
    Clients.Client_Name,                                -- Showing Client Names  from Clients table
    Properties.Monthly_rent AS Highest_rent             -- Showing the highest amount of rent
FROM
    RentInfo
JOIN                                                    -- Linking RentInfo table to Clients table by Client_No in both tables
    Clients ON RentInfo.Client_No = Clients.Client_No
JOIN                                                    -- Linking RentInfo table to Properties table by Property_No in both tables
    Properties ON RentInfo.Property_No = Properties.Property_No
WHERE                                                   -- Finding the highest paying rent from properties using MAX by filtering Monthly_rent
    Properties.Monthly_Rent = (
        SELECT MAX(Monthly_rent)
        FROM Properties);
```

Result 9

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Client_No	Client_Name	Highest_rent
CR-15	Gabriel Whitaker	2400

10. List all properties with rent amounts greater than the average rent amount across all properties.

Query 10.1

```
#10. List all properties with rent amounts greater than the average rent amount across all properties.
SELECT                                                    -- Finding the average monthly rent
    AVG(Monthly_rent)
FROM
    Properties;
```

Result 10.1

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

AVG(Monthly_rent)
1700.0000

Query 10.2

```
SELECT                                          -- Selecting properties with rents that has greater amount than average
    Properties.Property_No,                     -- Showing Property number from Properties table
    Properties.Property_Address,                -- Showing Property Address from Properties table
    Properties.Monthly_rent                     -- Showing Monthly_rent from Properties table
FROM
    Properties
WHERE                                           -- Find the property information that has more than average monthly rent
    Properties.Monthly_rent > (
        SELECT
            AVG(Monthly_rent)
            FROM
                Properties);
```

Result 10.2

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Property_No	Property_Address	Monthly_rent
PG009	9 South Circular Road, Dublin	1800
PG010	10 South Circular Road, Dublin	1900
PG011	11 South Circular Road, Dublin	2000
PG012	12 South Circular Road, Dublin	2100
PG013	13 South Circular Road, Dublin	2200
PG014	14 South Circular Road, Dublin	2300
PG015	15 South Circular Road, Dublin	2400

Critical analysis

Note: In this section, I will explain how each queries are executed and what were the challenges, what insights I've gained during the process, and critical analysis of each results in the real estate industry.

1. Retrieve all clients along with their associated properties

To execute this query, JOIN has been used to link Clients, RentInfo and Properties tables. This query creates the relationship between clients and their rented properties. I've used 2 options to make sure the query works correctly. First option displays Client_No, Client_Name, Property_No, Property_Address, and Monthly_rent information. Using RentInfo table and joining Clients and Properties tables by their numbers. In the next option, only Client_Name and Property_Address has been displayed due to the fact that it would be more readable and easy to comprehend the correlation between the data.

This query gives the overview relationship between client and properties, providing rental activity, revealing which clients rented which properties. It's helpful to understand client preferences for location or the property, giving insightful information for the popular properties and recognizing the patterns in customer behaviour for their choices. But if we add more specific information like payments status, rental duration, or rental frequency, it would've been more specific to do a forecast on the most frequently rented properties or the longest duration for that property for each customer which will help predict the customer engagement and financial performance.

2. List all properties rented out by all clients whose name begins with 'D'

In second query's execution, JOIN and WHERE clause, LIKE operator has been used to find the clients with the name that start with the letter 'D'. I have also 2 options and in the first one, Client_name_D, Property_No, and Property_Address has been shown. In the WHERE clause, finding D letter in the first letter of the name by LIKE using 'D%' was the main part. The second option shows Property_No and Property_Address. The reason for using 2 options is to prove that the query works by finding letter D and showing the client names by that. Listing all properties rented by clients whose name starts with 'D' can help discover patterns or trends within specific segments such as finding the locations of that specific group. However, finding only letter D seems a little bit narrowed. Because if we change the query to a specific name or add more behavioural data could've been more useful to use in the real estate market.

3. List all clients who have properties rented out for a specific duration

This query provides rental logs of clients in RentInfo table using specific date range. JOIN clause help connect Clients and RentInfo tables and WHERE clause help discover the ones who has rented during that specific range. It can give the data of when the properties are rented most or least and signals the turning point for that periods. However, if it's only 2023-02-20 to 2023-10-20, there might not be any information since there can be no one who rented during that period. So it would be easy to analyse if it was seasonal or yearly since every business reports on monthly, seasonal, or yearly basis.

4. Calculate the total monthly rent for each client

To calculate the total monthly rent, using SUM was the best option. Additionally, joining Clients, Properties, and RentInfo tables to list all the customer's total monthly rent amount and grouping by Client_No and Client_name since it's aggregated query. This information is significant if the aim was to find who contributes most in a monthly basis and how to keep their loyalty. But, to make it more valuable, we can add trends aligned with time or rent amount to client satisfaction.

5. Find the owner of a specific property

In this query, the Property_No PG015 has been chosen and it joins Properties and Owners tables to find the owner of that property. The simple = symbol has been used to find the matching owner. This information is useful for finding the owner of a specific property, maybe some client made a complaint or wanted to report something. It would be easy to find the owner. But, the contact information of the owner should've

been included for a more detailed investigation. Because only with the owner name, the information can be useless if there's more duplicated names with that owner.

6. Count the total number of properties owned by each owner

For the execution of this query, COUNT and GROUP BY has been used to calculate each owner's total number of properties. By joining Properties and Owners table and displaying Owner_No, Owner_Name, and Total_Properties_Count. The result can be utilized to find who has the most or least amount of properties. And maybe the real estate can offer more services or loyalty to them. However, it can also be risky if one owner owns more than fifty percent of the properties. Because, relying on only one owner could lead to future risks like losing all the estates or owner manipulates the market by increasing the overall rent price.

7. Identify owners who own multiple properties

To identify owners who has multiple properties, HAVING clause has been added by counting more than 1 property. Also GROUP BY is used since it's aggregated query. So this query shows the Owner_No, Owner_Name, Total_Properties_Count but deducts that owners who has only 1 property. This information provides the main owner who needs specific attention. However, it could be more useful if there was the types of the property to find the differences between owners.

8. List all clients along with the total rent they pay annually, sorted in ascending order

By using simple math equition of multiplication of 12 and ORDER BY ASC to see the lowest to top rent amounts annualy, this query is built. The query displays Client_No, Client_Name, and Annual_rent by joining Clients, Properties, RentInfo tables. Also GROUP BY Clients is used. This data helps discover the clients who pays less or more to revenue, which will help to find our focus on upsell or preservation. Even though most businesses like real estate signifies financial improvements and this tool is useful for that, the results should also include client loyalty and satisfaction.

9. Find the client who pays the highest monthly rent

This query uses subquery to find the maximum monthly rent in the WHERE clause. It joins Clients, Properties, and RentInfo tables and displays Client_No, Client_Name, and Highest rent which will result in giving only one customer information. This information is helpful if it's used in marketing and business incentives like giving rewards for those who contributes most and keeping the in the business longer. While finding the highest paying clients is good for identification, the real estate businesses shouldn't solely focus on one customer but to provide diversion in clients. If the diversion gap is too long, it can be a problem for the business too.

10. List all properties with rent amounts greater than the average rent amount across all properties

To execute this query, also subquery for finding the average rent is used in WHERE clause. Only Properties table is used and it displays Property_No, Property_Address, Monthly_rent which has more than the average value. To check if the query is working correctly, I've also created a query to find only the average value of the rent. The usefulness of this query is identifying expensive properties that can be marketed into different segment. It can also be helpful to target high value estates, and yet company should also include affordable properties for broader and longer market.

References

Generator, R. W., 2024. *Random Name Generator.* [Online]
Available at: https://randomwordgenerator.com/name.php